Océ European collaboration

Lou Dohmen <lajd@oce.nl>

Multidisciplinary requirements engineering

Printing for Professionals

MOOSE Workpackages

- WP1: Framework Development and Validation
- WP2: Systems & Software Requirements Engineering
  - Case 7: Multidisciplinary requirements engineering
  - Case 8: Req. engineering versus embedded software architecture (SAAM)
- WP3: Embedded Product Architectures
  - Case 9: The Caribou project (GORE, QAD and SBA)
  - Case 10: Re-design of architectural component (RUP)
  - Case 22: PDF filter platform (ADD)
- WP4: Improvement Management

Content

- Case
  - Case objective
  - Product
  - Experiments
    - Introduction
    - Execution
    - Results
  - Case lessons learned
- MOOSE lessons learned
- Benefits European collaboration

Case Objective

- Improve and experiment with the requirements engineering process in a multidisciplinary environment
  - Misalignment of software lifecycle and mechanical lifecycle
  - Lack of consistency in stakeholder needs
  - Insufficient system requirements documentation

Product

- The carrier is a sub-system (function) in a document reproduction system
- Goal of this function is collecting and finishing (e.g. stapling) a set of sheets
- The development of this finisher is in an early phase. After many discussions the technical concept has just been chosen.

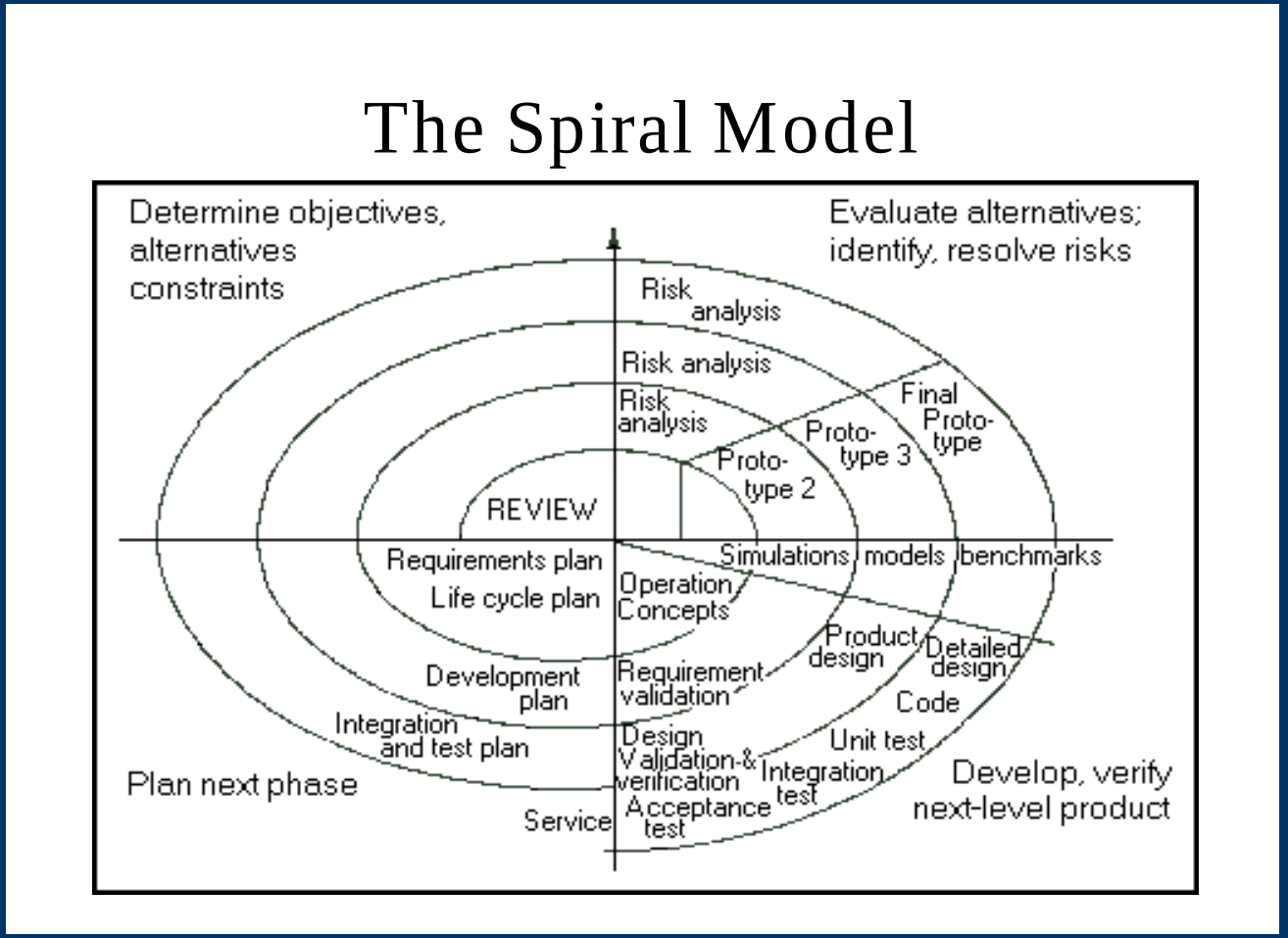
Product example 1

Experiments

- Misalignment of software lifecycle and mechanical lifecycle
- Lack of consistency in stakeholder needs
- Insufficient system requirements documentation
- Spiral model (B. Boehm, IEEE 1988)
- Gap-analysis between stakeholder needs and system requirements
- Use of DOORS
  - Document structure
  - Volere requirements template

Spiral model: Introduction

- A Spiral model of software development and enhancements, B. Boehm, IEEE, 1988

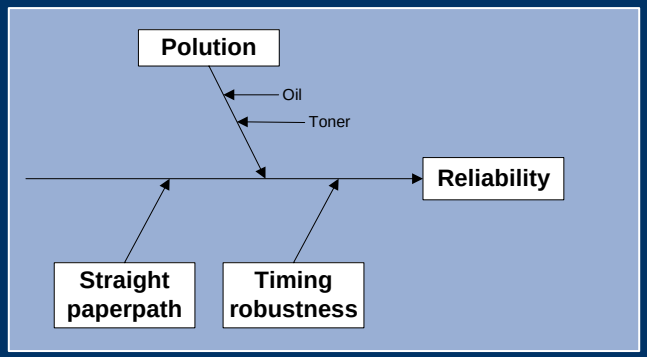


Spiral model: Introduction

- Similar to current multidisciplinary way of working within Océ
  - Several phases with formal milestone passing
  - Definition of phases based on risk-reduction
- Spiral model is a Meta-model
  - Techniques within cycles have to be defined
  - Approaches depend on amount of risk

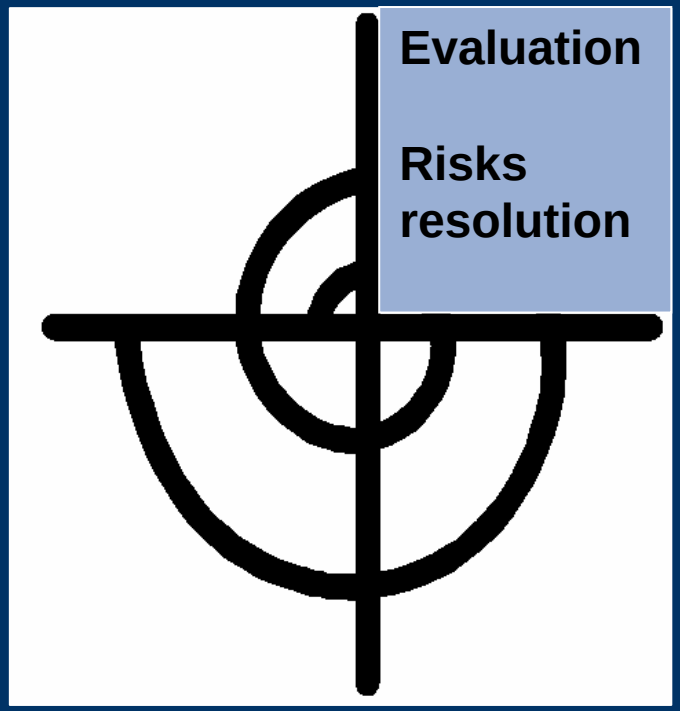
Spiral model: Execution

- Objectives
  - Reliability
  - Quality
  - Productivity
  - Modularity

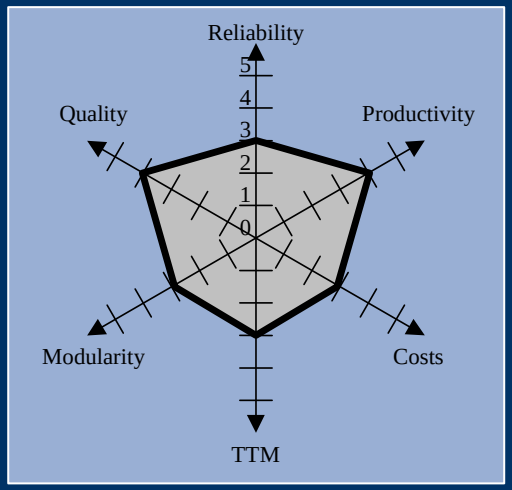


- Constraints
  - Standards (DFA)
  - Time-To-Market
  - Costs
- Alternatives

Spiral model: Execution



- Evaluate alternatives
  - Use of spider-diagrams (Kiviat diagram)

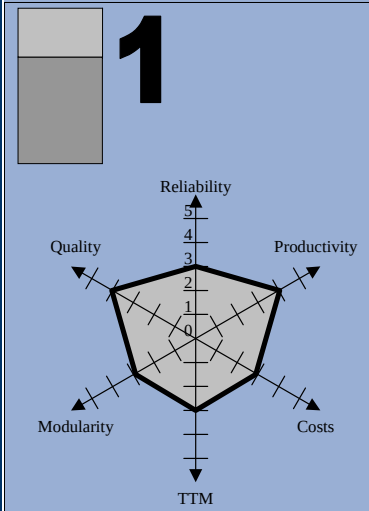


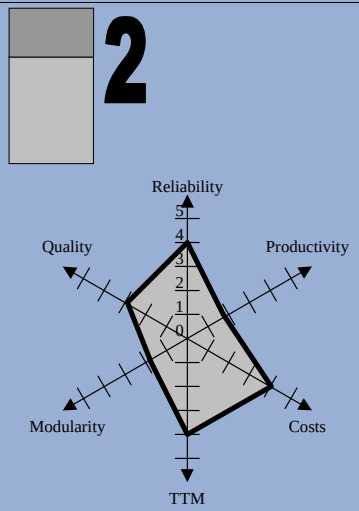
  - Distinguishing requirements
- Identify and resolve risks (for all alternatives!)

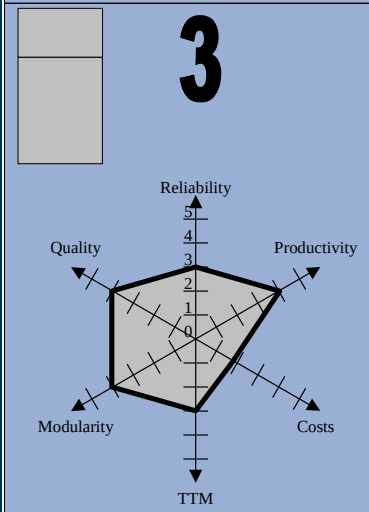
Evaluate alternatives

Distinguishing requirements are:

REQ 1.
REQ 2.
Impl. Constraint 1:







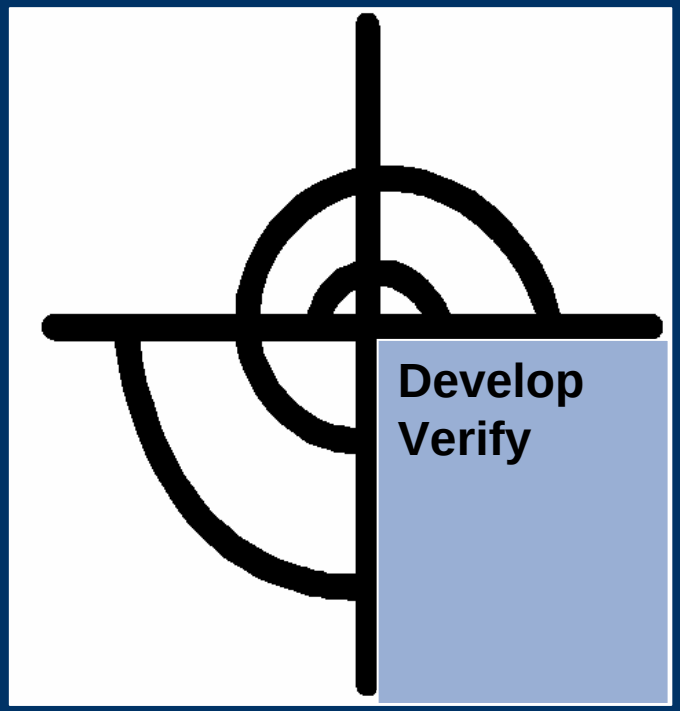
Rationale

Alternative 1 and 3 equally fulfil the objectives quality, reliability and productivity. Alternative 2 scores better on reliability (less complex) but less on quality productivity. Alternative 2 cannot process any yyy (REQ 1), see implementation constraint 1.
→ Alternative 2 is rejected

Alternative 1 and 3 differ in costs and modularity. Alternative 3 is more expensive, but more modular Alternative 3 requires more yyy (REQ 2) for the complete configuration.
→ Alternative 3 is rejected

Alternative 1 is chosen for the global finisher layout.

Spiral model: Execution

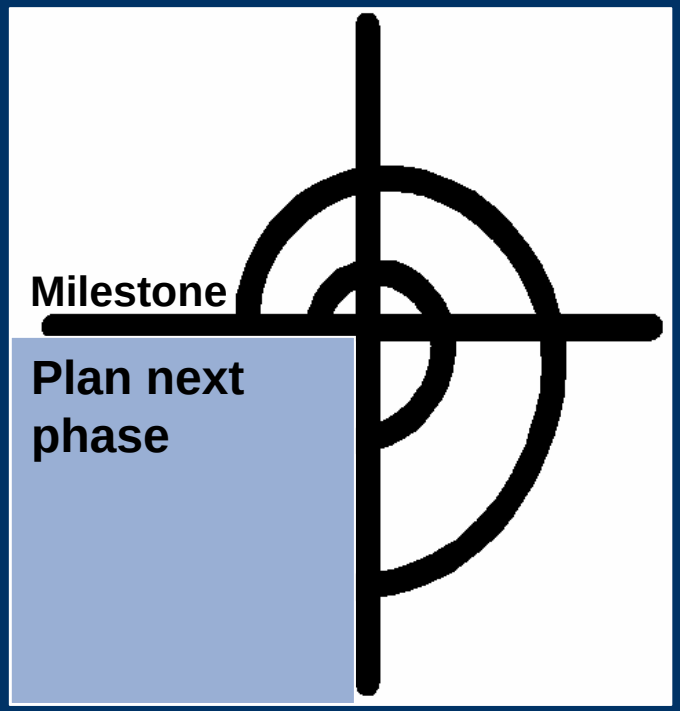


- Develop, verify next-level product (Risk mitigation)
  - Choose method / technology that best helps to resolve the risks
  - Simulation
  - Prototype
  - Timing models
  - Architectural models
  - Etc…

Spiral model: Execution

- Plan next phase
- Get commitment for next phase

- Milestone passing



Spiral model: Results

- Experiment was not finished because of external (political) decisions
- The results of the first iteration (cycle) were not valid anymore
- No time / commitment to start over again

- However, tailoring and experiences so far are valuable for future usage
- Internal report “Rationale finisher concept” based on CASE results

Gap-analysis: Introduction

- Good system requirements are the basis for successful function development
- Measurement of gap between system requirements and stakeholder needs (PSD)

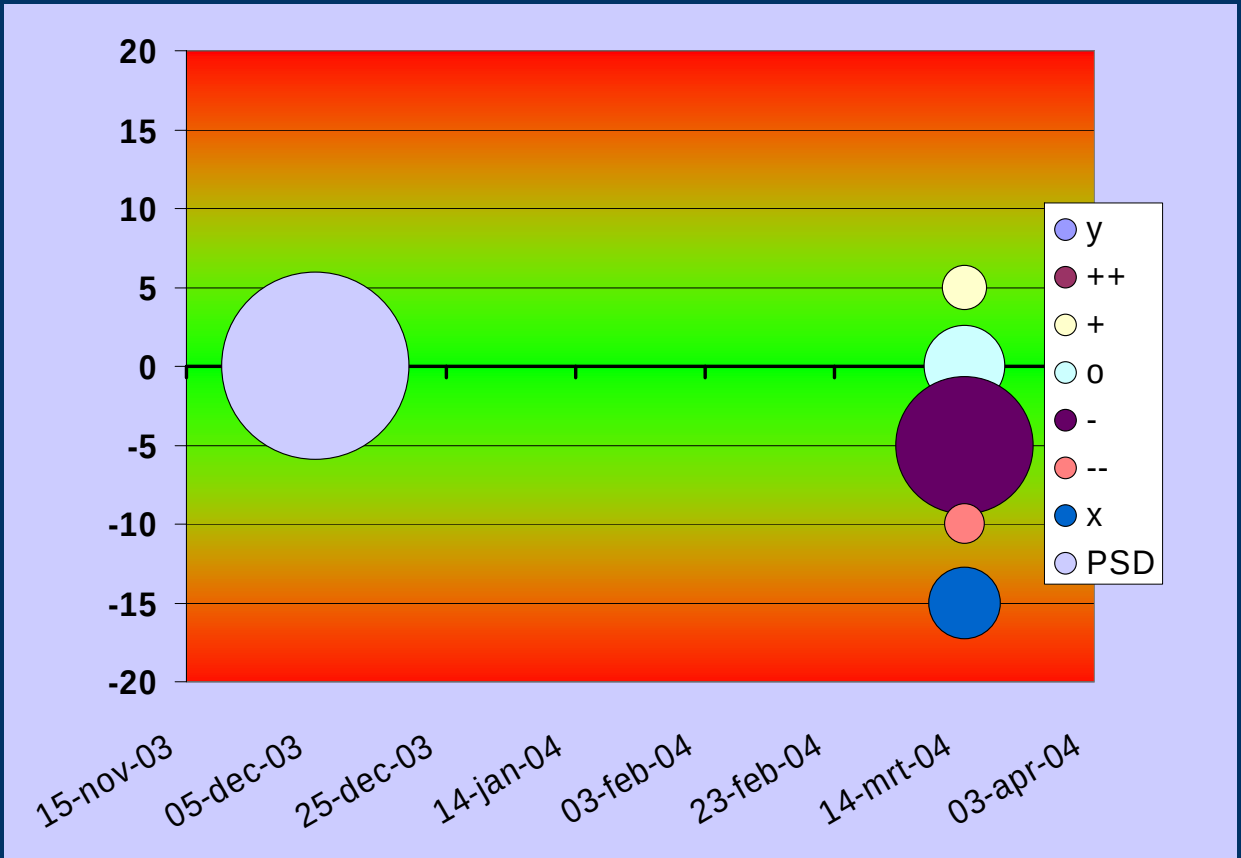
Gap-analysis: Execution

- < x > PSD requirements not matched in finisher
- < o > Fully covered
- Partly covered
  - < -- > significantly less functionality
  - < - > less functionality
  - < + > more functionality
  - < ++ > significantly more functionality
- < y > Finisher requirements not matched in PSD

Outcome	Number	Percentage
++	0	0
+	4	6
o	12	19
-	36	55
--	3	5
x	10	15
y	0	0

Gap-analysis: Results

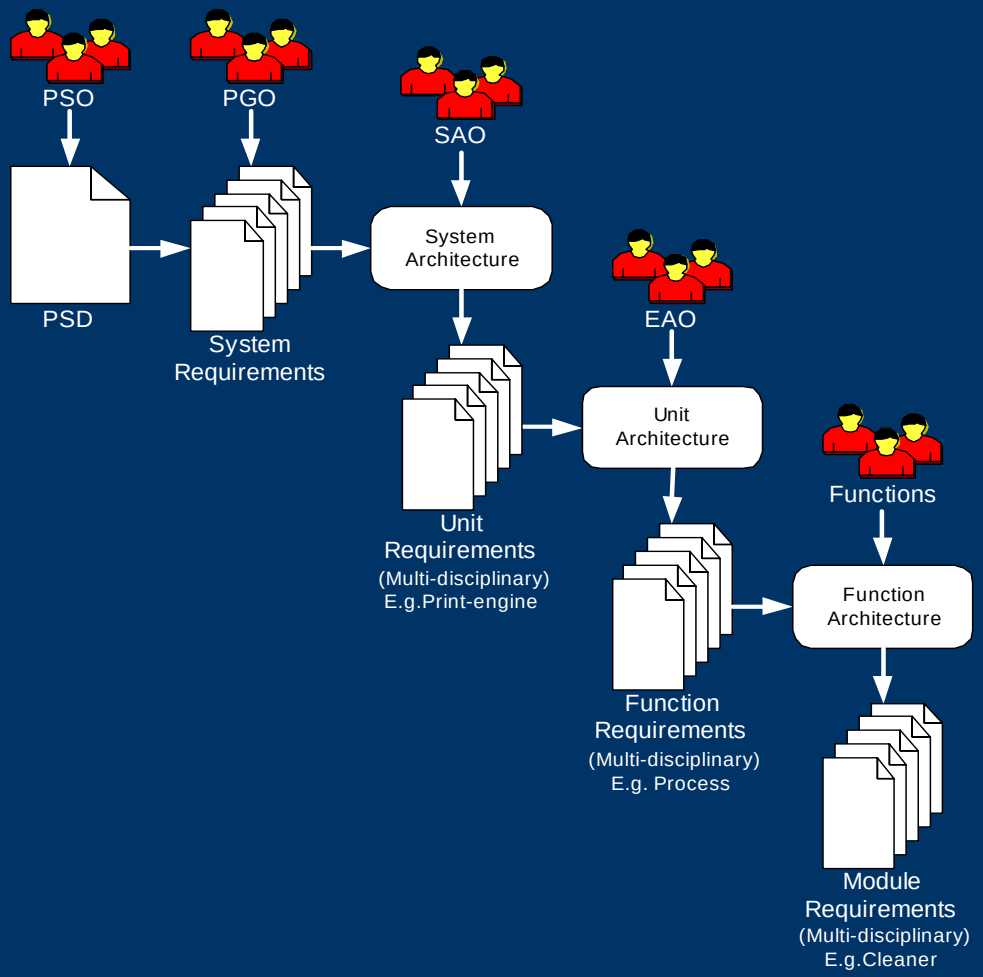
- 18% PSD requirements is not covered
- 78% Finisher requirements is only partly covered (70% will deliver less functionality)



DOORS: Introduction

- Why DOORS?
  - Alternative (requisite pro) already used within Océ
  - Integration with change-synergy
- Before starting
  - Define document structure
  - Define requirements template

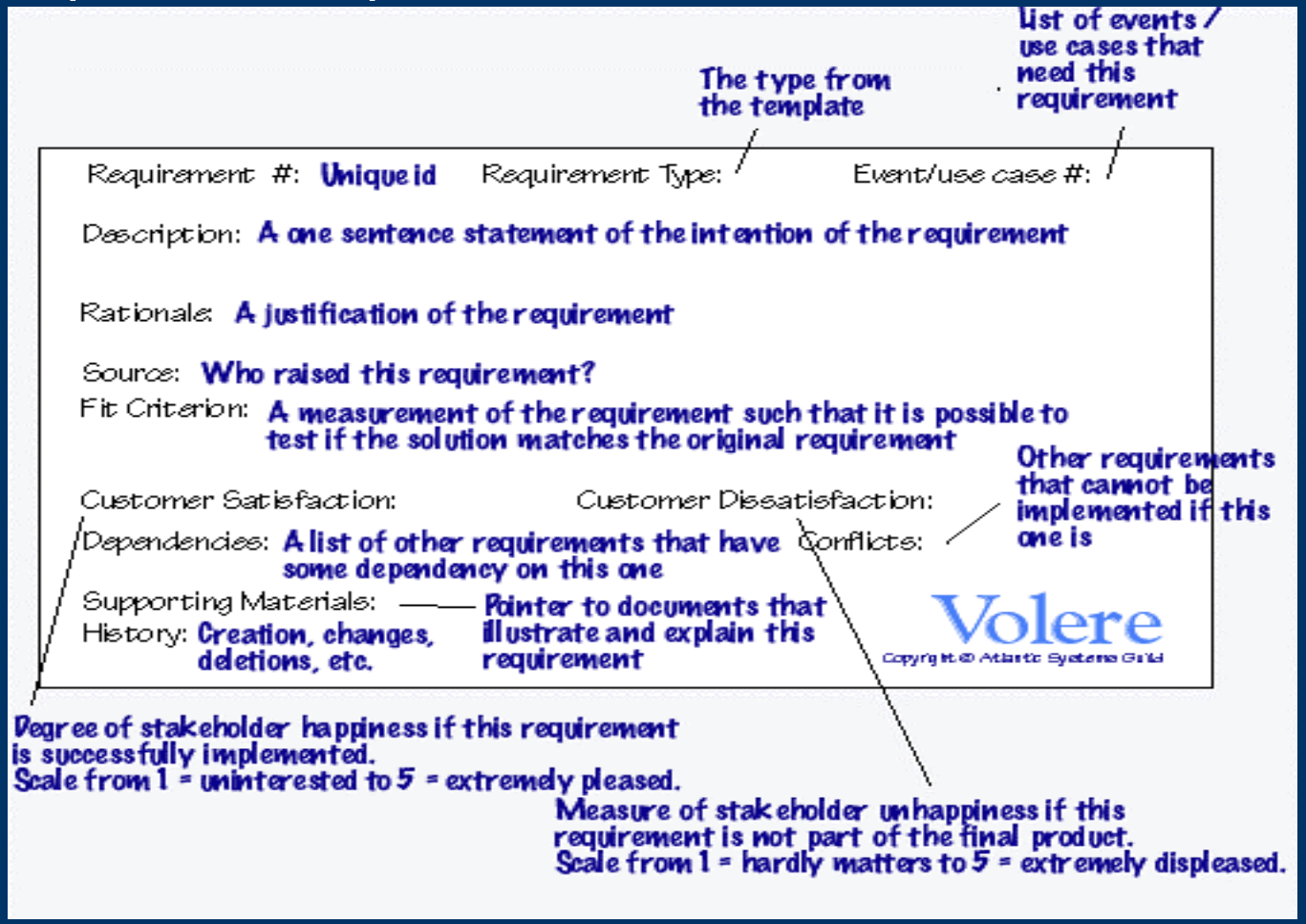
DOORS: Execution (Document structure)



- This is NOT a life-cycle (No V-model!)

DOORS: Execution (requirements template)

- S. Robertson and J. Robertson, “Mastering the requirements process”, 1999



DOORS: Execution (requirements template)

- Discarded attributes
  - Event/use case#: Not defined in the carrier project.
  - Rationale: Implemented by a link to the design
  - Customer (Dis)satisfaction: Not relevant during the piloting
  - Dependencies: N.a.
  - Supporting material: N.a.
  - Conflicts: N.a.
- Added attributes
  - Status (Draft, Proposed, Reviewed, Agreed)
  - Implementation certainty (Very high, high, low, very low)

Req. id:	**Req. type:**	**Status:**
		Source:
Synopsis:		
Details:		
Verification:		
		Impl. certainty:
History:		

DOORS: Results

- Added value RM-tool (DOORS)
  - Enabling efficient use of traceability
  - One entry and access point for all requirements
  - Requirements management based on individual requirements instead of document level
  - Efficient management of large set of requirements through user-defined views

Case lessons learned

- Changing / aligning lifecycles is a very tough and long process
- Objective quantification of requirements coverage contributes to meeting stakeholder expectation
- Good requirements management for a large set of requirements need tool support
- A document structure enables many discussions
  - What documents do we need?
  - When are the documents written?
  - Who will write the document?
  - What is the content of a document?

MOOSE lessons learned

- A real project as carrier has severe disadvantages:
  - competing priorities; project priorities always win
  - coupling in time
  - many external forces (politics, resources, ..)
- Advantage real project: real success story instead of a succeeded experiment
- Getting commitment for executing a case-study may take some time (several months)
  - Keep commitment by short term successes
- Explicitly allocate resources to the project; 10% - rules do not work! (AIO's, students)
- Clearly define the objectives / expectations (funding, knowledge, collaboration, ..)

Benefits European collaboration

- Personal expectations for collaboration were too high!
  - Low (mainly NL) contribution to cases
  - Geographic / culture barrier (especially Spain)
  - ??
- Collaborations fulfils overall project (MOOSE) goals
  - Inventory, Tailoring, Cases, ….
- Networking: meet people with similar problems/challenges (mainly NL)
- ‘Benchmarking’ of Océ product development
- ‘Space’ to explore new technologies (effort/attention/priority)
- 2 new Océ-employees (AIO’s)

References

- Barry Boehm, “A spiral model of software development and enhancement”, IEEE, 1988
- Barry Boehm, “Spiral development: Experiences, Principles, and Refinements”, CEI, July 2000
- Suzanne Robertson and James Robertson, “Mastering the requirements process”, 1999
- www.telelogic.com